REMARKS

Claims 1-15 are pending in the above-identified patent application. Claims 1, 7, 10 and 13 are independent claims.

The examiner requested that a comma be inserted after the word "executed" appearing on line 3 of claim 1. Applicant amended claim 1 accordingly.

The examiner objected to claim 2 on the ground that the claim uses acronyms without first defining them. In response, applicant amended claim 2 to recite the features previously recited in claim 3, and to further recite the meaning of the syntax "br=ctx" and "br!=ctx". Applicant also canceled claim 3, and amended claim 4 to correct its dependency.

The examiner rejected claim 6 under 35 U.S.C. §112, second paragraph, on the ground that the term "the branch operation" does not have an antecedent basis. Applicant amended claim 6 to recite "a branch operation".

The examiner rejected claims 1-9 and 13-15 under 35 U.S.C. §101 on the ground that the invention is directed to a non-statutory subject matter.

Specifically, with respect to claim 1, the examiner stated:

> 8. Regarding claim 1, the claim is non-statutory for two reasons:
> a) the claim calls for "a context branch instruction that, when executed, causes a data processing apparatus to cause an instruction stream to branch...". This language does not include a positive recitation that the branch actually happens. For instance, with a conditional branch, a preceding evaluation of a condition, by itself, ultimately causes the branch instruction to actually branch. However, the evaluation itself is not a tangible result. The examiner recommends replacing "cause an instruction stream to branch to another instruction of the instruction stream" with - branch to another instruction of an instruction stream-. This way, the data processing apparatus will be understood to actually perform a branch as opposed to just performing a condition evaluation which causes a branch, which does not produce a tangible result.
> b) for the claimed context branch instruction, branching will occur in accordance with an evaluation, but when the branch does not occur, all that occurs is the evaluation, which is not a tangible result. For instance, take the case of applicant's context branch instruction being the "br=ctx[ctx#.. " instruction. For this instruction, an evaluation will be made as to whether ctx# equals a current context number. If the numbers match, then a branch will occur. But, if a match does not occur, then nothing happens in addition to the evaluation. Consequently, since evaluating does not produce a tangible result, the claim as a whole does not produce a tangible result. It follows that claim 1 is non-statutory for this reason. (Office Action, page 3, paragraph 8)

In response, applicant amended independent claim 1 to clarify that execution of applicant's branch instruction causes the apparatus to select another instruction in the instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction and an instruction following the context branch instruction, and to recite that the selected instruction is retrieved. Applicant similarly amended independent claims 7, 10 and 13. Thus, applicant's independent claim 1 provides that in both circumstances (i.e., if the branch condition is satisfied, and if the branch condition is not satisfied), execution of the context branch instruction results in the selection of an instruction. In other words, if the branch condition is fulfilled, one instruction will be selected and retrieved, and if it is not fulfilled a second instruction will be selected and retrieved. Accordingly, applicant submits that the features recited in independent claim 1 produce a tangible result, and that claim 1 is directed to statutory subject matter.

Because independent claim 1 produces a tangible result and is directed to statutory subject matter, claims 2-6, which depend from claim 1 also produce tangible results, and are thus directed to statutory subject matter.

Independent claims 7, 10 and 13, and the claims that depend from them, are also directed to statutory subject matter for reasons similar to those provided with respect to independent claim 1.

The examiner rejected claims 1, 7-9 and 13-15 under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,314,510 to Saulsbury. Further, the examiner rejected claims 2-6 under 35 U.S.C. §103(a) as being unpatentable over Saulsbury in view of U.S. Patent No. 6,564,316 to Peters. The examiner also rejected claims 10-12 under 35 U.S.C. §103(a) as being unpatentable over Saulsbury.

Specifically, regarding claim 1, the examiner stated:

> 26. Referring to claim 1, Saulsbury has taught a computer program product residing on a computer readable storage medium (Fig.2, component 102; column 2, lines 42-45) comprising instructions (Fig.4), including a context branch instruction (Fig.4 and column 4, lines 44-45; note the "branch on zero" (bz) instruction) that, when executed, causes a data processing apparatus to cause an instruction stream to branch to another instruction of the instruction stream associated with a label specified by the context branch instruction (Fig.4; note that the "bz next" instruction will branch to the instruction associated with label "next", which is the "blst wrl, dbl" instruction) based on an evaluation of whether

> a current context number matches a context number specified by the context branch instruction. The bz instruction is a branch on zero instruction, which means that if the context number specified by the branch instruction (zero) matches a current context number (i.e., the value to be compared to zero), then a branch will occur. In this program, the "bz next#" instruction checks to see if the current context number (dirty bit db0) is equal to 0, the specified context number. See column 4, lines 39-45. It should be noted that the branch is a context branch and the numbers are context numbers as the branches and numbers are associated with contexts (i.e., executing environments). See the title, abstract, and column 2, lines 40-41. (Office Action, pages 7-8, paragraph 26)

Applicant amended independent claim 1 to clarify that the selection of one of the branch target instruction and the instruction following the context branch instruction is based on a comparison of the current executing context number and the context number specified by the context branch instruction. Applicant similarly amended independent claims 7, 10 and 13. Applicant also amended claims 8, 11 and 14, which depend from independent claims 7, 10, and 13 respectively, to make the language recited in claims 8, 11 and 14 consistent with the amended language of independent claims 7, 10 and 13.

Applicant's independent claim 1 recites "a context branch instruction that, when executed, causes a data processing apparatus to: select another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction and an instruction following the context branch instruction based on a comparison of a current executing context number to a context number specified by the context branch instruction; and retrieve the selected other instruction." Thus, applicant's context branch instruction causes both the performance of a comparison operation, and the retrieval of the next instruction to be executed. Furthermore, the comparison operation is performed on the current executing context number and the context number value specified by the context branch instruction itself.

In contrast, Saulsbury describes a microprocessor that has a reduced context switching overhead for handling traps (col. 1, lines 6-8). To that end, Saulsbury describes a program and a trap handling routine. Specifically, Saulsbury explains:

> **After the modified nop instruction is executed, a trap may occur. In this case, the microprocessor 100 transfers execution from the program to a conventional trap handling routine. The write special register instruction**

> **wrspr in the trap handling routine causes the operand stored by the working register wr1 to be saved in a special register sp0 of the special register file 112. The write dirty bit registers instruction wrdbr then causes the dirty bits stored by the dirty bit registers dbr0 to dbrN-1 to be stored in the working register wr1. Once this has been done, the bit test instruction btst determines if the dirty bit stored by the dirty bit register dbr0 is zero or one. If the dirty bit is zero, this indicates that the operand stored by the working register wr0 is inactive. In this case, the trap handling routine branches to the bit test instruction btst at the label next1 as a result of the branch on zero instruction bz. Thus, the operand stored by the working register wr0 is not saved to the main memory 104 because it has become inactive. But, if the dirty bit is one, then this indicates that the operand is active. In this case, the trap handling routine does not branch and the store instruction st causes the operand stored in the working register wr0 to be saved in the main memory 104 at the address addr2. (emphasis added, col. 4, lines 30-51)**

While Saulsbury does not specifically describe what operations are performed in the course of executing a "branch on zero" instruction, conventionally, the execution of such an instruction requires that the executing microprocessor determine if the zero-flag of the ALU is set to 0. In the example illustrated in Saulsbury's FIG. 4, the zero-flag will presumably be set as a result of execution of the instruction preceding the "branch on zero instruction", namely, "btst wr1, db0", which "determines if the dirty bit stored by the dirty bit register dbr0 is zero or one" (col. 4, lines 40-41).

Thus, unlike applicant's independent claim 1, a comparison operation (if one is even performed by Saulsbury's execution of the "btst wr1, db0" instruction) is performed as a result of the instruction preceding Saulsbury's branch instruction, and not as a result of execution of the branch instruction itself.

Moreover, such a comparison operation would compare the content of two registers and not the current executing context number, as provided by the microprocessor, to a context number value specified by the branch instruction. Indeed, Saulsbury's branch instruction does not specify any type of value. Accordingly, Saulsbury fails to disclose or suggest at least "a context branch instruction that, when executed, causes a data processing apparatus to: select another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction and a sequential instruction following the context branch instruction based on a comparison of a current context number provided by the data processing apparatus and a context number specified by the context branch instruction; and

retrieve the selected other instruction," as required by applicant's independent claim 1. Applicant's independent claim 1 is therefore patentable over the cited art.

Claims 2 and 4-6 depend from independent claim 1 and are therefore patentable for at least the same reasons as independent claim 1.

Independent claims 7, 10 and 13 recite "performing a comparison of a context number of an executing context to a context number specified by a context branch instruction; selecting another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction and an instruction following the context branch instruction based on the comparison; and retrieving the selected other instruction," or similar language. For reasons similar to those provided with respect to independent claim 1, at least these features are not disclosed by the cited art. Applicant's independent claims 7, 10 and 13 are therefore patentable over the cited art.

Claims 8-9 depend from independent claim 7 and are therefore patentable for at least the same reasons as independent claim 7. Claims 11-12 depend from independent claim 10 and are therefore patentable for at least the same reasons as independent claim 10. Claims 14-15 depend from independent claim 13 and are therefore patentable for at least the same reasons as independent claim 13.

All of the dependent claims are patentable for at least the reasons for which the claims on which they depend are patentable.

In view of the foregoing, applicant respectfully submits that the application is in condition for allowance and such action is respectfully requested at the examiner's earliest convenience.

Canceled claims, if any, have been canceled without prejudice or disclaimer.

Any circumstance in which the applicant has (a) addressed certain comments of the examiner does not mean that the applicant concedes other comments of the examiner, (b) made arguments for the patentability of some claims does not mean that there are not other good reasons for patentability of those claims and other claims, or (c) amended or canceled a claim does not mean that the applicant concedes any of the examiner's positions with respect to that claim or other claims.

No fee is believed due. Please apply any other charges to deposit account 06-1050, referencing attorney docket 10559-307US1.

Respectfully submitted,

Date: 3/20/07

Denis G. Maloney
Attorney for Intel Corporation
Reg. No. 29,670

Customer No. 20985
Fish & Richardson P.C.
Telephone: (617) 542-5070
Facsimile: (617) 542-8906

21586698.doc